Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 4

Тема Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.

Студент Никуленко И.В.

Группа ИУ7-42Б

Оценка (баллы) _______________

Преподаватель Градов В.М.

Москва.
2021 г

Цель работы. Получение навыков построения алгоритма метода наименьших квадратов с использованием полинома заданной степени при аппроксимации табличных функций с весами.

1 Исходные данные

1. Таблица функции с весами ρ_i с количеством узлов N. Сформировать таблицу самостоятельно со случайным разбросом точек.
2. Степень аппроксимирующего полинома - n.

2 Код программы

Код программы представлен на листингах 1-2.

Листинг 1. func.py

```
def coef_finder(matrix, n):
    coefficients = []
    for i in range(n):
        summ = matrix[i][n]
        for j in range(len(coefficients)):
            summ -= coefficients[j] * matrix[i][j]
        coefficients.append(summ)
    return coefficients


def root_mean_square(table, n):
    n += 1
    matrix = [[0] * (n + 1) for i in range(n)]
    for i in range(n):
        for j in range(i, n):
            summ = 0
            for k in range(len(table)):
                summ += table[k][2] * table[k][0] ** (i + j)
            matrix[i][j] = matrix[j][i] = summ

        summ = 0
        for k in range(len(table)):
            summ += table[k][2] * table[k][1] * table[k][0] ** i
        matrix[i][n] = summ

    for i in range(n - 1, 0, -1):
        tmp = matrix[i][i]
        for j in range(n + 1):
            matrix[i][j] /= tmp

        for j in range(i):
            tmp = matrix[j][i]
            for k in range(n + 1):
                matrix[j][k] -= matrix[i][k] * tmp
    matrix[0][n] /= matrix[0][0]
```

```
    coefficients = coef_finder(matrix, n)

    dots = []
    x = table[0][0]
    while x <= table[len(table) - 1][0]:
        y = 0
        for j in range(len(coefficients)):
            y += coefficients[j] * x ** j
        dots.append(y)
        x += 0.1

    return dots
```

Листинг 2. main.py

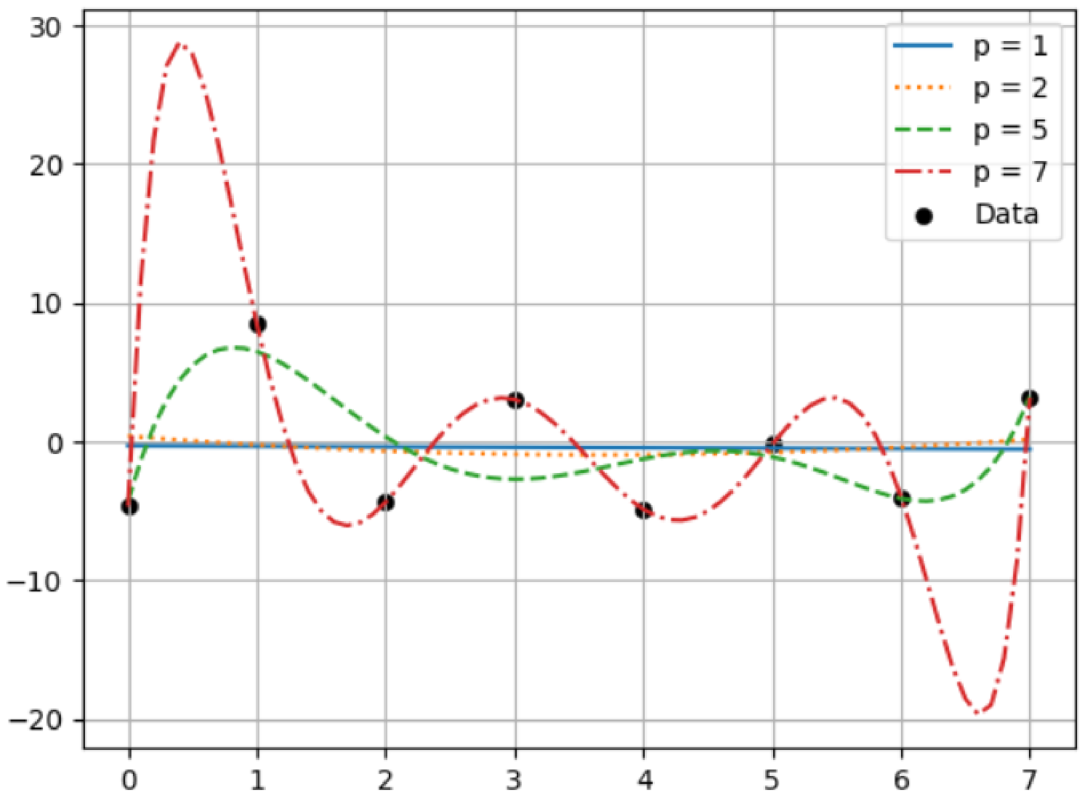
```
from random import uniform, randint
import matplotlib.pyplot as plt
from func import root_mean_square
from numpy import array



def print_func(table):
    table = array(table)
    dots = [[]]
    x = table[0][0]
    while x <= table[len(table) - 1][0]:
        dots[0].append(x)
        x += 0.1

    for i in (1, 2, 5, 7):
        dots.append(root_mean_square(table, i))

    funcs = ['p = 1', 'p = 2', 'p = 5', 'p = 7', 'Data']

    fig, ax = plt.subplots()

    style = ["solid", "dotted", "dashed", "-."]
    for i in range(1,5):
        ax.plot(dots[0], dots[i], linestyle=style[i - 1])

    ax.scatter([i[0] for i in table], [i[1] for i in table],
               c='black', linewidths=0.25)

    plt.legend(funcs, loc=1)
    plt.grid()

    plt.show()
```

```python
def print_comparsion(table):
    table = array(table)
    dots = [[]]
    x = table[0][0]
    while x <= table[len(table) - 1][0]:
        dots[0].append(x)
        x += 0.1

    for i in range(1,3):
        dots.append(root_mean_square(table, i))

    for i in range(len(table)):
        table[i][2] = 1

    for i in range(1,3):
        dots.append(root_mean_square(table, i))

    funcs = ['p = 1 (различный вес точек)',
        'p = 2 (различный вес точек)',
        'p = 1 (одинаковый вес точек)',
        'p = 2 (одинаковый вес точек)']

    fig, ax = plt.subplots()
    style = ["solid", "dotted", "dashed", "--"]
    for i in range(1,5):
        ax.plot(dots[0], dots[i], linestyle=style[i - 1])

    ax.scatter([i[0] for i in table], [i[1] for i in table], c='black')
    plt.legend(funcs, loc=1)
    plt.grid()
    plt.show()


def main():
    table = [[i, uniform(-10,10), randint(1,30)] for i in range(8)]

    print('Таблица функции:\n')
    print("+{:3s}|{:5s}|{:5s}|{:3s}+".format('-', '-', '-', '-').replace(' ', '-'))
    print('| № |  X  |  Y  |Вес|')
    print("|{:3s}|{:5s}|{:5s}|{:3s}|".format('-', '-', '-', '-').replace(' ', '-'))
    for i in range(len(table)):
        print('|{:3d}|{:5d}|{:5.1f}|{:3d}|'.format(i + 1, table[i][0], table[i][1], table[i][2]))
        print("|{:3s}|{:5s}|{:5s}|{:3s}|".format('-', '-', '-', '-').replace(' ', '-'))

    print_func(table)
    print_comparsion(table)


if __name__ == "__main__":
    main()
```

3 Результаты работы

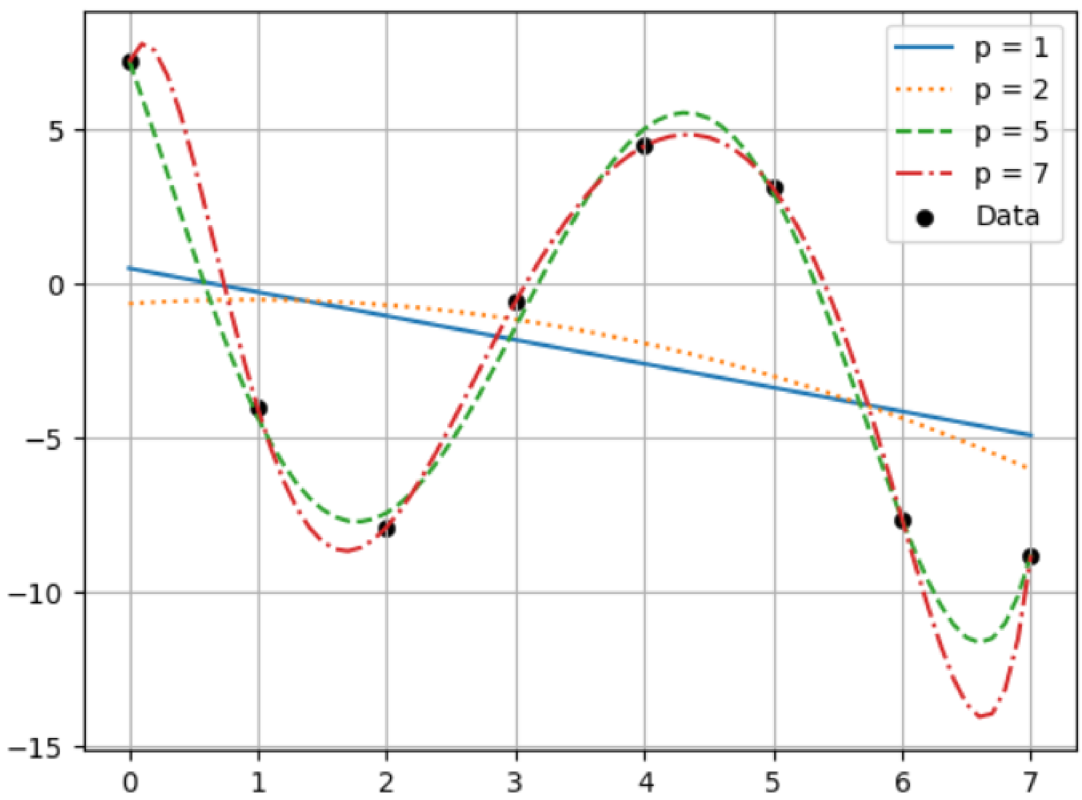
1. Веса всех точек одинаковы.

№	X	Y	Вес
1	0	-4.7	1
2	1	8.5	1
3	2	-4.4	1
4	3	3.0	1
5	4	-4.9	1
6	5	-0.2	1
7	6	-4.0	1
8	7	3.2	1

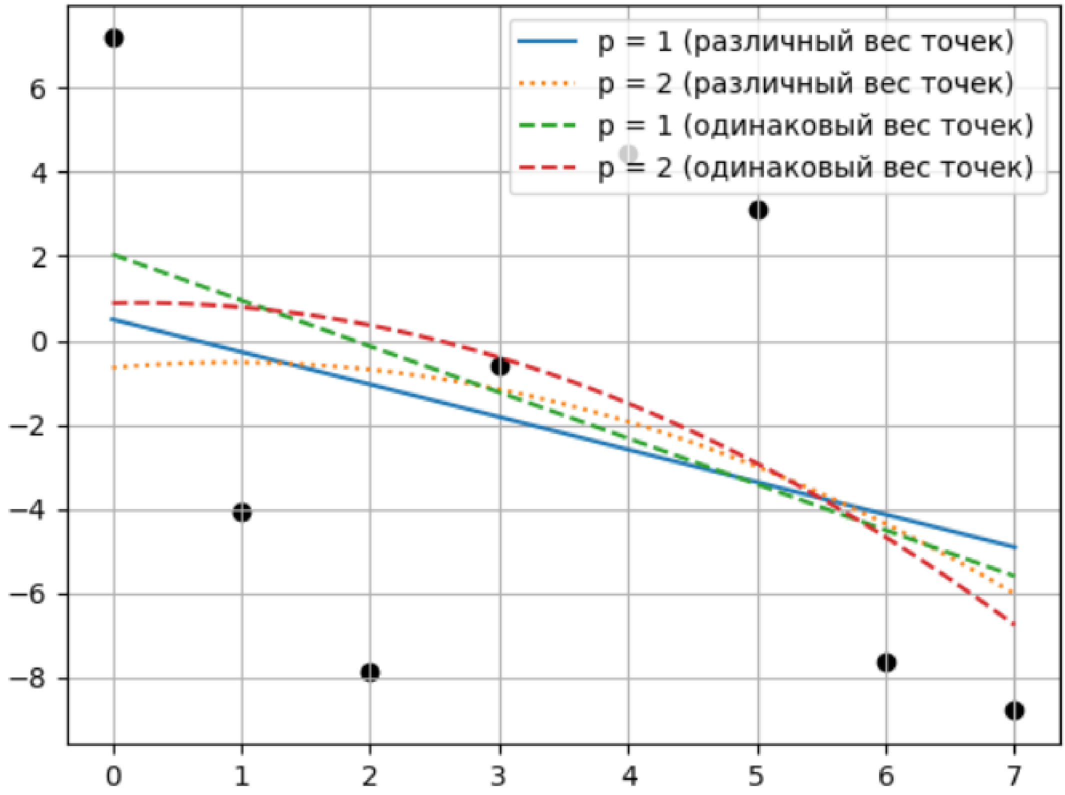


2. Веса точек разные.

№	X	Y	Вес
1	0	7.2	16
2	1	-4.0	20
3	2	-7.9	30
4	3	-0.6	21
5	4	4.5	20
6	5	3.1	21
7	6	-7.6	19
8	7	-8.8	16



Изменение знака углового коэффициента прямой с помощью изменения веса:



4 Вопросы при защите лабораторной работы

1. Что произойдет при задании степени полинома n = N-1 (числу узлов таблицы минус 1)?

Полином будет построен так, что его график будет проходить через все точки, так как для однозначного определения полинома N-1 степени достаточно N точек.

2. Будет ли работать Ваша программа при n >= N ? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?

Программа будет работать, но результат работы программы будет некорректным, так как с момента когда n станет равным N определитель системы линейных уравнений будет равен 0. В таком случае есть вероятность возникновения аварийной остановки при приведении диагональной матрицы к единичной, где может произойти деление на ноль.

3. Получить формулу для коэффициента полинома a_0 при степени полинома n=0. Какой смысл имеет величина, которую представляет данный коэффициент?

$$a_0 = \Sigma^{N}_{i=1} p_i y_i / \Sigma^{N}_{i=1} p_i$$

Данный коэффициент является взвешенным средним арифметическим ординат функции.

4. Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда n=N=2. Принять все $\rho_i = 1$.

$$\begin{cases} a_0 + (x_1 + x_2)a_1 + (x_1^2 + x_2^2)a_2 = y_1 + y_2 \\ (x_1 + x_2)a_0 + (x_1^2 + x_2^2)a_1 + (x_1^3 + x_2^3)a_2 = y_1x_1 + y_2x_2 \\ (x_1^2 + x_2^2)a_0 + (x_1^3 + x_2^3)a_1 + (x_1^4 + x_2^4)a_2 = y_1x_1^2 + y_2x_1^2 \end{cases}$$

Определитель данной СЛАУ равен 0, система не имеет решений.

5. Построить СЛАУ при выборочном задании степеней аргумента полинома , $\varphi(x) = a_0 + a_1 x^m + a_2 x^n$ причем степени n и m в этой формуле известны.

$$\begin{cases} (x^0,\ x^0)a_0 + (x^0,\ x^m)a_1 + (x^0,\ x^n)a_2 = (y, x^0) \\ (x^m,\ x^n)a_0 + (x^m,\ x^m)a_1 + (x^m,\ x^n)a_2 = (y, x^m) \\ (x^n,\ x^0)a_0 + (x^n,\ x^m)a_1 + (x^n,\ x^n)a_2 = (y, x^n) \end{cases}$$

6. Предложить схему алгоритма решения задачи из вопроса 5, если степени n и m подлежат определению наравне с коэффициентами a_k , т.е. количество неизвестных равно.

Перебрать все возможные пары n и m, для каждой пары вычислить все коэффициенты a_i и ошибку. Результат — пара с минимальной ошибкой.